IN THE CLAIMS

1 (Currently Amended). A method comprising:

placing an executable thread of instructions in an inactive state in response to a resource being unavailable; ~~and~~

when the resource becomes available, changing the thread of instructions to the active state and granting the resource to the thread of instructions[[.]]; and

releasing the thread, in response to the thread finishing with the resource, only if the thread is not dependent on any subsequent thread.

Claim 2 (Canceled).

3 (Previously Presented). The method of claim 1 further comprising executing the thread of instructions when in the active state.

Claims 4 and 5 (Canceled).

6 (Original). The method of claim 1 further comprising maintaining an indication of a state of each of a plurality of executable threads of instructions.

7 (Original). The method of claim 6 wherein the indication of the state of each thread comprises a state variable corresponding to a dependency, if any, of an associated thread.

Claims 8-10 (Canceled).

11 (Currently Amended). An apparatus comprising:

an execution circuit to receive and execute a thread of instructions, wherein the execution circuit transmits a semaphore request message and places the thread in an inactive state in response to the thread of instructions requiring a resource having an associated semaphore; and

a semaphore entity coupled with the execution circuit to receive the semaphore request message from the execution circuit and to selectively grant control of the semaphore in

response to the semaphore request message by transmitting a semaphore acknowledge message to the execution circuitry, wherein the execution circuitry, in response to receiving the semaphore acknowledge message, removes the thread of instructions from the inactive state and grants the resource to the thread when the resource becomes available[[.]], said execution entity to release the thread in response to the thread finishing with the resource only if the thread is not dependent on any subsequent threads.

12 (Original). The apparatus of claim 11 further comprising: at least one additional execution circuit to execute threads of instructions; and a thread dispatcher coupled with the execution circuit and at least one additional execution circuit to dispatch threads for execution by selected execution circuits.

13 (Original). The apparatus of claim 11, wherein the execution circuitry, in response to receiving the semaphore acknowledge message, resumes execution of the thread of instructions including accessing the resource associated with the semaphore.

14 (Original). The apparatus of claim 11 wherein when the thread of instructions is in the inactive state, execution of the instructions ceases and the execution circuitry does not poll the semaphore entity to determine a status of the semaphore request message.

15 (Currently Amended). An system comprising:

a memory controller; and

an execution circuit coupled with the memory controller to receive and execute a thread of instructions, wherein the execution circuit transmits a request message and places the thread in an inactive state in response to the thread of instructions requiring a resource that is unavailable, said execution unit to automatically change the thread of instructions to an active state and grant the resource to the thread of instructions when the resource becomes available[[.]], said execution entity to release the thread in response to the thread finishing with the resource only if the thread is not dependent on any subsequent threads.

16 (Original). The system of claim 15 further comprising: at least one additional execution circuit to execute threads of instructions; and a thread dispatcher coupled with the execution circuit and at least one additional execution circuit to dispatch threads for execution by selected execution circuits.

17 (Currently Amended). The system of claim 15, wherein the execution circuitry, in response to receiving a [[the]] semaphore acknowledge message, resumes execution of the thread of instructions including accessing the resource associated with the semaphore.

18 (Original). The system of claim 15 wherein when the thread of instructions is in the inactive state, execution of the instructions ceases and the execution circuitry does not poll the semaphore entity to determine a status of the semaphore request message.

19 (Previously Presented). The method of claim 1 including placing requests for a semaphore in a queue.

Claim 20 (Cancel).

21 (Currently Amended). The method of claim 19 [[20]] including automatically granting the resource to the thread whose request is the next request in the queue.